**图解 SQL 里的各种 JOIN**

[mzlogin](https://www.zhihu.com/people/mzlogin)[mzlogin](https://www.zhihu.com/people/mzlogin)

2 小时前

从业以来主要在做客户端，用到的数据库都是表结构比较简单的 SQLite，以我那还给老师一大半的 SQL 水平倒也能对付。现在偶尔需要到后台的 SQL Server 里追查一些数据问题，就显得有点捉襟见肘了，特别是各种 JOIN，有时候傻傻分不清楚，于是索性弄明白并做个记录。

**前言**

在各种问答社区里谈及 SQL 里的各种 JOIN 之间的区别时，最被广为引用的是 CodeProject 上[C.L. Moffatt](http://link.zhihu.com/?target=https%3A//www.codeproject.com/script/Membership/View.aspx%3Fmid%3D5909363) 的文章 [Visual Representation of SQL Joins](http://link.zhihu.com/?target=https%3A//www.codeproject.com/Articles/33052/Visual-Representation-of-SQL-Joins)，他确实讲得简单明了，使用文氏图来帮助理解，效果明显。本文将沿用他的讲解方式，稍有演绎，可以视为该文较为粗糙的中译版。

**约定**

下文将使用两个数据库表 Table\_A 和 Table\_B 来进行示例讲解，其结构与数据分别如下：

mysql> SELECT \* FROM Table\_A ORDER BY PK ASC;

+----+------------+

| PK | Value |

+----+------------+

| 1 | FOX |

| 2 | COP |

| 3 | TAXI |

| 4 | LINCION |

| 5 | ARIZONA |

| 6 | WASHINGTON |

| 7 | DELL |

| 10 | LUCENT |

+----+------------+

8 rows in set (0.00 sec)

mysql> SELECT \* from Table\_B ORDER BY PK ASC;

+----+-----------+

| PK | Value |

+----+-----------+

| 1 | TROT |

| 2 | CAR |

| 3 | CAB |

| 6 | MONUMENT |

| 7 | PC |

| 8 | MICROSOFT |

| 9 | APPLE |

| 11 | SCOTCH |

+----+-----------+

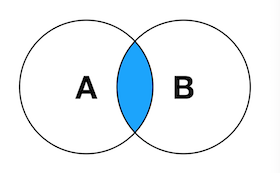
8 rows in set (0.00 sec)

**常用的 JOIN**

**INNER JOIN**

INNER JOIN 一般被译作内连接。内连接查询能将左表（表 A）和右表（表 B）中能关联起来的数据连接后返回。

**文氏图：**



**示例查询：**

SELECT A.PK AS A\_PK, B.PK AS B\_PK,

A.Value AS A\_Value, B.Value AS B\_Value

FROM Table\_A A

INNER JOIN Table\_B B

ON A.PK = B.PK;

查询结果：

+------+------+------------+----------+

| A\_PK | B\_PK | A\_Value | B\_Value |

+------+------+------------+----------+

| 1 | 1 | FOX | TROT |

| 2 | 2 | COP | CAR |

| 3 | 3 | TAXI | CAB |

| 6 | 6 | WASHINGTON | MONUMENT |

| 7 | 7 | DELL | PC |

+------+------+------------+----------+

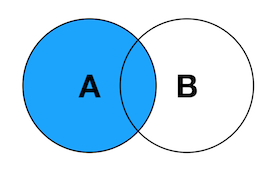
5 rows in set (0.00 sec)

*注：其中* *A* *为* *Table\_A* *的别名，B* *为* *Table\_B* *的别名，下同。*

**LEFT JOIN**

LEFT JOIN 一般被译作左连接，也写作 LEFT OUTER JOIN。左连接查询会返回左表（表 A）中所有记录，不管右表（表 B）中有没有关联的数据。在右表中找到的关联数据列也会被一起返回。

**文氏图：**



**示例查询：**

SELECT A.PK AS A\_PK, B.PK AS B\_PK,

A.Value AS A\_Value, B.Value AS B\_Value

FROM Table\_A A

LEFT JOIN Table\_B B

ON A.PK = B.PK;

查询结果：

+------+------+------------+----------+

| A\_PK | B\_PK | A\_Value | B\_Value |

+------+------+------------+----------+

| 1 | 1 | FOX | TROT |

| 2 | 2 | COP | CAR |

| 3 | 3 | TAXI | CAB |

| 4 | NULL | LINCION | NULL |

| 5 | NULL | ARIZONA | NULL |

| 6 | 6 | WASHINGTON | MONUMENT |

| 7 | 7 | DELL | PC |

| 10 | NULL | LUCENT | NULL |

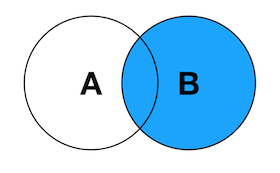
+------+------+------------+----------+

8 rows in set (0.00 sec)

**RIGHT JOIN**

RIGHT JOIN 一般被译作右连接，也写作 RIGHT OUTER JOIN。右连接查询会返回右表（表 B）中所有记录，不管左表（表 A）中有没有关联的数据。在左表中找到的关联数据列也会被一起返回。

**文氏图：**



**示例查询：**

SELECT A.PK AS A\_PK, B.PK AS B\_PK,

A.Value AS A\_Value, B.Value AS B\_Value

FROM Table\_A A

RIGHT JOIN Table\_B B

ON A.PK = B.PK;

查询结果：

+------+------+------------+-----------+

| A\_PK | B\_PK | A\_Value | B\_Value |

+------+------+------------+-----------+

| 1 | 1 | FOX | TROT |

| 2 | 2 | COP | CAR |

| 3 | 3 | TAXI | CAB |

| 6 | 6 | WASHINGTON | MONUMENT |

| 7 | 7 | DELL | PC |

| NULL | 8 | NULL | MICROSOFT |

| NULL | 9 | NULL | APPLE |

| NULL | 11 | NULL | SCOTCH |

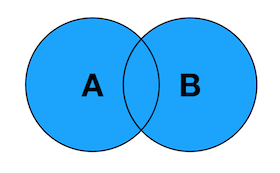
+------+------+------------+-----------+

8 rows in set (0.00 sec)

**FULL OUTER JOIN**

FULL OUTER JOIN 一般被译作外连接、全连接，实际查询语句中可以写作 FULL OUTER JOIN 或 FULL JOIN。外连接查询能返回左右表里的所有记录，其中左右表里能关联起来的记录被连接后返回。

**文氏图：**



**示例查询：**

SELECT A.PK AS A\_PK, B.PK AS B\_PK,

A.Value AS A\_Value, B.Value AS B\_Value

FROM Table\_A A

FULL OUTER JOIN Table\_B B

ON A.PK = B.PK;

查询结果：

ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'FULL OUTER JOIN Table\_B B

ON A.PK = B.PK' at line 4

*注：我当前示例使用的 MySQL 不支持* *FULL OUTER JOIN。*

应当返回的结果（使用 UNION 模拟）：

mysql> SELECT \*

-> FROM Table\_A

-> LEFT JOIN Table\_B

-> ON Table\_A.PK = Table\_B.PK

-> UNION ALL

-> SELECT \*

-> FROM Table\_A

-> RIGHT JOIN Table\_B

-> ON Table\_A.PK = Table\_B.PK

-> WHERE Table\_A.PK IS NULL;

+------+------------+------+-----------+

| PK | Value | PK | Value |

+------+------------+------+-----------+

| 1 | FOX | 1 | TROT |

| 2 | COP | 2 | CAR |

| 3 | TAXI | 3 | CAB |

| 4 | LINCION | NULL | NULL |

| 5 | ARIZONA | NULL | NULL |

| 6 | WASHINGTON | 6 | MONUMENT |

| 7 | DELL | 7 | PC |

| 10 | LUCENT | NULL | NULL |

| NULL | NULL | 8 | MICROSOFT |

| NULL | NULL | 9 | APPLE |

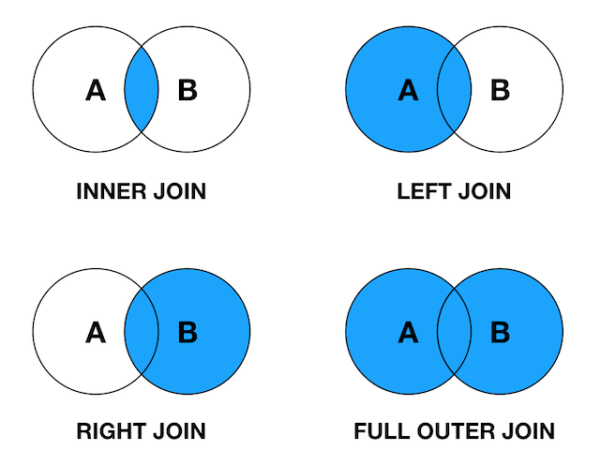
| NULL | NULL | 11 | SCOTCH |

+------+------------+------+-----------+

11 rows in set (0.00 sec)

**小结**

以上四种，就是 SQL 里常见 JOIN 的种类和概念了，看一下它们的合影：



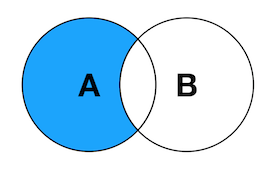
有没有感觉少了些什么，学数学集合时完全不止这几种情况？确实如此，继续看。

**延伸用法**

**LEFT JOIN EXCLUDING INNER JOIN**

返回左表有但右表没有关联数据的记录集。

**文氏图：**



**示例查询：**

SELECT A.PK AS A\_PK, B.PK AS B\_PK,

A.Value AS A\_Value, B.Value AS B\_Value

FROM Table\_A A

LEFT JOIN Table\_B B

ON A.PK = B.PK

WHERE B.PK IS NULL;

查询结果：

+------+------+---------+---------+

| A\_PK | B\_PK | A\_Value | B\_Value |

+------+------+---------+---------+

| 4 | NULL | LINCION | NULL |

| 5 | NULL | ARIZONA | NULL |

| 10 | NULL | LUCENT | NULL |

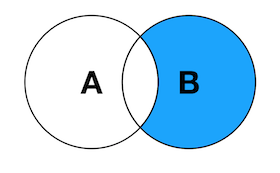
+------+------+---------+---------+

3 rows in set (0.00 sec)

**RIGHT JOIN EXCLUDING INNER JOIN**

返回右表有但左表没有关联数据的记录集。

**文氏图：**



**示例查询：**

SELECT A.PK AS A\_PK, B.PK AS B\_PK,

A.Value AS A\_Value, B.Value AS B\_Value

FROM Table\_A A

RIGHT JOIN Table\_B B

ON A.PK = B.PK

WHERE A.PK IS NULL;

查询结果：

+------+------+---------+-----------+

| A\_PK | B\_PK | A\_Value | B\_Value |

+------+------+---------+-----------+

| NULL | 8 | NULL | MICROSOFT |

| NULL | 9 | NULL | APPLE |

| NULL | 11 | NULL | SCOTCH |

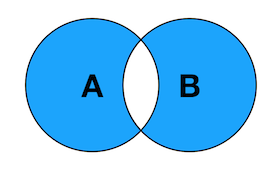
+------+------+---------+-----------+

3 rows in set (0.00 sec)

**FULL OUTER JOIN EXCLUDING INNER JOIN**

返回左表和右表里没有相互关联的记录集。

**文氏图：**



**示例查询：**

SELECT A.PK AS A\_PK, B.PK AS B\_PK,

A.Value AS A\_Value, B.Value AS B\_Value

FROM Table\_A A

FULL OUTER JOIN Table\_B B

ON A.PK = B.PK

WHERE A.PK IS NULL

OR B.PK IS NULL;

因为使用到了 FULL OUTER JOIN，MySQL 在执行该查询时再次报错。

ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'FULL OUTER JOIN Table\_B B

ON A.PK = B.PK

WHERE A.PK IS NULL

OR B.PK IS NULL' at line 4

应当返回的结果（用 UNION 模拟）：

mysql> SELECT \*

-> FROM Table\_A

-> LEFT JOIN Table\_B

-> ON Table\_A.PK = Table\_B.PK

-> WHERE Table\_B.PK IS NULL

-> UNION ALL

-> SELECT \*

-> FROM Table\_A

-> RIGHT JOIN Table\_B

-> ON Table\_A.PK = Table\_B.PK

-> WHERE Table\_A.PK IS NULL;

+------+---------+------+-----------+

| PK | Value | PK | Value |

+------+---------+------+-----------+

| 4 | LINCION | NULL | NULL |

| 5 | ARIZONA | NULL | NULL |

| 10 | LUCENT | NULL | NULL |

| NULL | NULL | 8 | MICROSOFT |

| NULL | NULL | 9 | APPLE |

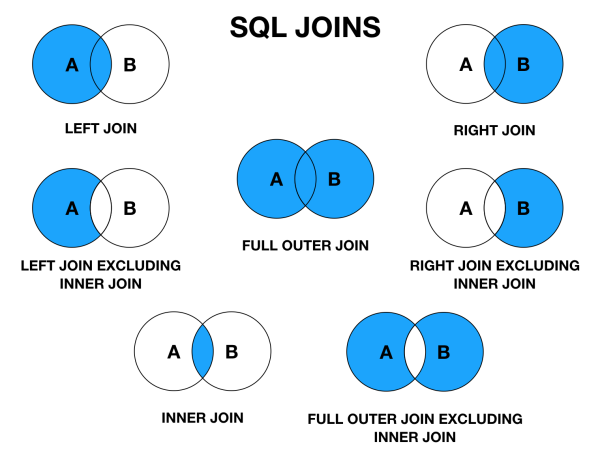
| NULL | NULL | 11 | SCOTCH |

+------+---------+------+-----------+

6 rows in set (0.00 sec)

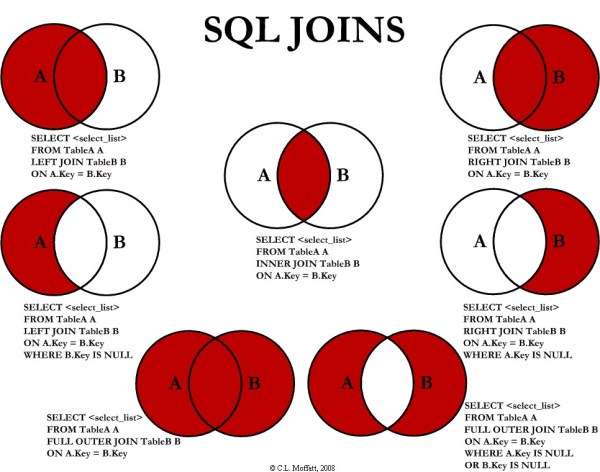
**总结**

以上七种用法基本上可以覆盖各种 JOIN 查询了。七种用法的全家福：



看着它们，我仿佛回到了当年学数学，求交集并集的时代……

顺带张贴一下 [C.L. Moffatt](http://link.zhihu.com/?target=https%3A//www.codeproject.com/script/Membership/View.aspx%3Fmid%3D5909363) 带 SQL 语句的图片，配合学习，风味更佳：



**补充说明**

1. 文中的图使用 Keynote 绘制；
2. 个人的体会是 SQL 里的 JOIN 查询与数学里的求交集、并集等很像；
3. SQLite 不支持 RIGHT JOIN 和 FULL OUTER JOIN，可以使用 LEFT JOIN 和 UNION 来达到相同的效果；
4. MySQL 不支持 FULL OUTER JOIN，可以使用 LEFT JOIN 和 UNION 来达到相同的效果；
5. 还有更多的 JOIN 用法，比如 CROSS JOIN（迪卡尔集）、SELF JOIN，目前我还未在实际应用中遇到过，且不太好用图来表示，所以并未在本文中进行讲解。如果需要，可以参考 [SQL JOINS Slide Presentation](http://link.zhihu.com/?target=https%3A//www.w3resource.com/slides/sql-joins-slide-presentation.php) 学习。

假如你对我的文章感兴趣，可以关注我的微信公众号 isprogrammer 随时阅读更多内容。

**参考**

* [Visual Representation of SQL Joins](http://link.zhihu.com/?target=https%3A//www.codeproject.com/Articles/33052/Visual-Representation-of-SQL-Joins)
* [How to do a FULL OUTER JOIN in MySQL?](http://link.zhihu.com/?target=https%3A//stackoverflow.com/questions/4796872/how-to-do-a-full-outer-join-in-mysql)
* [SQL JOINS Slide Presentation](http://link.zhihu.com/?target=https%3A//www.w3resource.com/slides/sql-joins-slide-presentation.php)